CSC3210 ASSIGNMENT-2

Attention：

An explanation is expected with every answer and may count for half of the points or more. If your answer does not have an explanation, or that explanation is incomplete, expect to lose points. This is because we want to see how you arrive at the answer.

1.) For the RISC-V assembly instructions below, what is the corresponding C statement? Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the arrays A and B are in registers x10 and x11, respectively. (Normally, we were working with int, so we used slli x5, 2. In this assignment, we are working with long, so we use slli x5, 3 instead)

[10 POINTS]

```
slli x30, x5, 3          // x30 = f*8
add x30, x10, x30        // x30 = &A[f]
slli x31, x6, 3          // x31 = g*8
add x31, x11, x31        // x31 = &B[g]
lw x5, 0(x30)            // f = A[f]

addi x12, x30, 8
lw x30, 0(x12)
add x30, x30, x5
lw x30, 0(x31)
```

2.) Assume that registers x5 and x6 hold the values 0x8000000000000000 and 0xD000000000000000, respectively. **[6*5=30 POINTS]**

NOTE: Overflow occurs when an arithmetic operation produces a result that exceeds the representable range of the destination register.

Here we use RISC-V 64 (RV64): the size of the register is 64-bits.

a) What is the value of x30 for the following assembly code?

add x30, x5, x6

b) Is the result in x30 the desired result, or has there been overflow?

c) For the contents of registers x5 and x6 as specified above, what is the value of x30 for the following assembly code?

sub x30, x5, x6

d) Is the result in x30 the desired result, or has there been overflow?

e) For the contents of registers x5 and x6 as specified above, what is the value of x30 for the following assembly code?

add x30, x5, x6
add x30, x30, x5

f) Is the result in x30 the desired result, or has there been overflow?

3.) Provide the instruction type and assembly language instruction for the following binary value:
$0000\ 0000\ 0001\ 0000\ 1000\ 0000\ 1011\ 0011_{two}$
Hint: The following figures may be helpful. **[10 POINTS]**

Name (Field Size)	Field 7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	Comments
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

Instruction	Format	funct7	rs2	rs1	funct3	rd	opcode
add (add)	R	0000000	reg	reg	000	reg	0110011
sub (sub)	R	0100000	reg	reg	000	reg	0110011
Instruction	**Format**	**immediate**		**rs1**	**funct3**	**rd**	**opcode**
addi (add immediate)	I	constant		reg	000	reg	0010011
lw (load word)	I	address		reg	010	reg	0000011
Instruction	**Format**	**immed -iate**	**rs2**	**rs1**	**funct3**	**immed -iate**	**opcode**
sw (store word)	S	address	reg	reg	010	address	0100011

FIGURE 2.5 RISC-V instruction encoding. In the table above, "reg" means a register number between 0 and 31 and "address" means a 12-bit address or constant. The funct3 and funct7 fields act as additional opcode fields.

Format	Instruction	Opcode	Funct3	Funct6/7
R-type	add	0110011	000	0000000
	sub	0110011	000	0100000
	sll	0110011	001	0000000
	xor	0110011	100	0000000
	srl	0110011	101	0000000
	sra	0110011	101	0000000
	or	0110011	110	0000000
	and	0110011	111	0000000
	lr.d	0110011	011	0001000
	sc.d	0110011	011	0001100
I-type	lb	0000011	000	n.a.
	lh	0000011	001	n.a.
	lw	0000011	010	n.a.
	lbu	0000011	100	n.a.
	lhu	0000011	101	n.a.
	addi	0010011	000	n.a.
	slli	0010011	001	000000
	xori	0010011	100	n.a.
	srli	0010011	101	000000
	srai	0010011	101	010000
	ori	0010011	110	n.a.
	andi	0010011	111	n.a.
	jalr	1100111	000	n.a.
S-type	sb	0100011	000	n.a.
	sh	0100011	001	n.a.
	sw	0100011	010	n.a.
SB-type	beq	1100111	000	n.a.
	bne	1100111	001	n.a.
	blt	1100111	100	n.a.
	bge	1100111	101	n.a.
	bltu	1100111	110	n.a.
	bgeu	1100111	111	n.a.
U-type	lui	0110111	n.a.	n.a.
UJ-type	jal	1101111	n.a.	n.a.

FIGURE 2.18 RISC-V instruction encoding. All instructions have an opcode field, and all formats except U-type use the funct3 field. R-type instructions use the funct7 field, and immediate shifts (slli, srli, srai) use the funct6 field.

4.) Provide the instruction type and hexadecimal representation of the following instruction:

sw x5, 32(x30) **[10 POINTS]**

5.) Assume x5 holds the value 0x0000000001010000. What is the value of x6 after the following instructions? (ORI: or Immediate) **[10 POINTS]**

NOTE: Here we use RISC-V 64 (RV64): the size of the register is 64-bits.

```
        bge x5, x0, ELSE
        jal x0, DONE
ELSE:   ori x6, x0, 2
DONE:
```

6.) Consider a proposed new instruction named rpt. This instruction combines a loop's condition check and counter decrement into a single instruction.
For example rpt x29, loop would do the following:

```
if (x29 > 0) {
    x29 = x29 −1;
    goto loop
}
```

a) If this instruction were to be added to the RISC-V instruction set, what is the most appropriate instruction format? **[10 POINTS]**

b) What is the shortest sequence of RISC-V instructions that performs the same operation? **[5 POINTS]**

7.)

a) Translate the following C code to RISC-V assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers x5, x6, x7, and x29, respectively. Also, assume that register x10 holds the base address of the array D. **[10 POINTS]**

```
for(i=0; i<a; i++)
        for(j=0; j<b; j++)
                D[4*j] = i + j;
```

b) How many RISC-V instructions does it take to implement the C code from (a)? If the variables a and b are initialized to 10 and 1 and all elements of D are initially 0, what is the total number of RISC-V instructions executed to complete the loop? **[5 POINTS]**